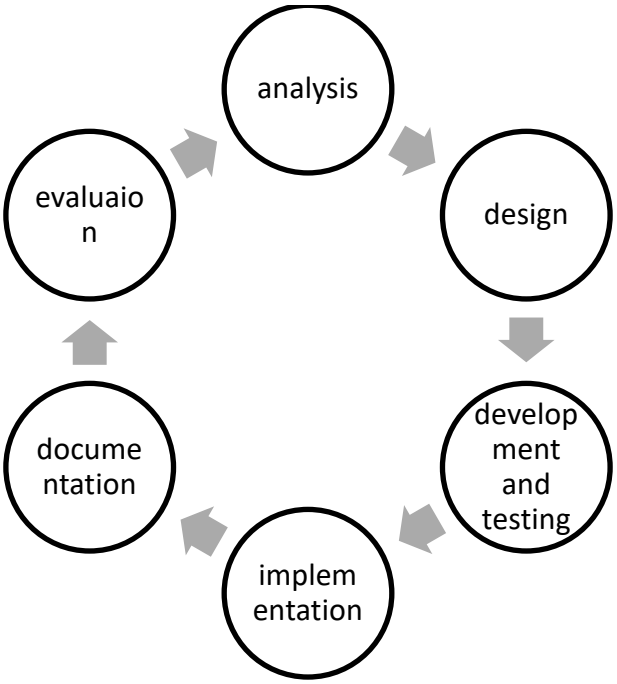
16 System Life Cycle

16.1 The stages in the system life cycle



- **analysis:** Collecting information about the present system and identifying problems
- **design:** Designing a new system to correct the problems identified in the analysis, it is when the design specification is produced
- **development and testing:** Developing and testing new system
- **implementation:** Replacing the old system with the new system
- **documentation:** Creating technical and user documentation for new system
- **evaluation:** Evaluating whether the new system meets the requirements of the design requirements

16.2 Analysis

Analysis involves finding out how the current system works and what the requirements of the client are for the new system

Methods of Research

Questionnaires:

- questionnaires are used when there is a large number of users making it a larger sample, conducting interviews becomes impractical and hence the questionnaires are used as the results can be quantified and compared
- they aren't suitable when there's a small number of users size since the sample size isn't large enough to gauge different onions and its much easier conduct their interviews rather than spending time creating an appropriate questionnaire
- an exception of this would be when there's a continuous clash between the user and analyzer to arrange an appointment
- the disadvantage of questionnaires is that it doesn't let the analyzer the opportunity to ask the users to elaborate on their answer without having to contact them again
- It's important for questions to be asked in a way in which the required information can be elicited from users and those responses can be analyzed and quantified collectively
- this could be done by providing appropriate multiple choices so that each response is counted
- the questionnaire shouldn't be long otherwise not many responses will be returned
- A mixture of multiple choice questions, opinion ratings and open questions should be used to allow a balance of quantitative analysis of closed questions and a qualitative analysis of open questions where users are able to suggest alternative ideas to those presented by the questionnaire
- Questions should also be written in a way which doesn't threaten the users and the way they currently do their work
- Users should also be given the opportunity to submit their questionnaires anonymously as it allows for more honest answers to be given
- It Is also ideal for them to be held online as it allows results to be immediately stored and readily available for detailed analysis in the form of graphs and tables
- Filters can be applied to the results and responses can be compared based on the answers given to another question

Interviews:

- Interviews involve a direct conversation between the client and analysis
- Where there is a single end user or small group of end users then interviews are the perfect solution because questions can be asked of the users and conversation can take place which can expand upon answers that are given with follow up questions searching for further detail
- In large organizations, interviews can be used with key stakeholders or representatives of user groups.
- Questions to be asked should be planned and designed to elicit the required information from the client, they could also vary depending on who is being interviewed. Eg; if management is, then questions will focus on requirements of the organization, if end users are being interviewed, then questions will be aimed at finding out what users need to make their jobs more efficient
- However it could be difficult to hold an interview due to unavailability of the clients
- Clients could also answer dishonestly as interviews tend not to be anonymous, and they are afraid they might be judged by their answers
- The analyst also has to be fully involved with each interview which could result in a lot of cost and time being drained up early in the project.

Observation:

- Observation involves the analyst watching the processes that take place within an organization to find out how everyday tasks are completed
- this can involve sitting with the users to understand the tasks they have to complete, with an opportunity to ask questions of the users to elicit further information that could be needed for a requirements specification
- one disadvantage of this method is that user might do things differently from normal or they be more efficient when they know they r being observed, hence this doesn't give a true picture of what is happening
- this method can take up a lot of time however it is the most insightful method of finding out how an organization works

Document analysis:

- Existing documents within an organization an tell an analyst a lot about the information that is currently being used
- The analyst will need to see examples of any documents that show output information or five an indication of what data is being collected for input to a system
- The analyst can sometimes also identify processes that take place by looking at documents
- Its also possible to estimate the amount of data that Is likely to be required if the volume of the documents is known
- This method is not to be used on its own but must be used in conjunction with other analysis methods because its difficult to identify the processes just by looking at the documents
- Examination of the documents also only shows data that is currently output and doesn't give the opportunity to find out what additional data an organization might need or not need
- This information can be found out by following up documentation analysis with interviews.

User Requirements Specification

- It's a contract between the developer and the client
- Its created by the analyst in consultation with the client who will approve it
- It specifies what the client needs the system to do so that developer can produce a system that meets the client's needs

Contents:

- The purpose of the system
- The main objectives of the system
- Data that must be output from the system (for example; invoices, sales report)
- Data that needs to be input to the system to generate the outputs, including any screens or data collection forms
- Validation and verification that is need for input data
- Processes that need to take place to convert inputs into outputs or to store data
- Data that need to be stored
- Functional requirements such as performance measures
- Deadlines for each milestone within the project

System Specification

- It is created by the designer
- It lists all the software and hardware that is needed for the new system
- Software needs are identified first as the hardware depends on what software needs
- Only software need to run the system should be specified, there may be different software identified for different types of users and servers
- Storage space required for data used by system is considered
- The analyst will probably recommend higher than minimum specifications so that the system functions at a reasonable speed
- These specifications will include processing power and the amount of memory required
- External hardware components needed should also be specified and these should be based upon the requirements of the user

Design Specification

- It is created by the designer and is an illustration of how the system will look like, what the data structure will be and how the system will work
- It is intended to give the user an idea of the system will look like before it is being developed so that the user feedback can be incorporated into the final designs,
- The developer will then follow the designs
- A design specification will also include house style(logos, colors, fonts, styles sizes), screen sizes, connectivity diagram to show the links between screens and purpose of calculations

16.3 Design

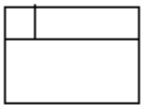
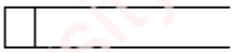
During the **design** stage, the overall structure of the system and details of the system components are designed without being developed.

Data Flow Diagram (DFD)

- DFD shows how data flows throughout the system
- It's not about the order of processes; it's purely about the data flows.
- It can exist at many levels. At level 0 or context level, the diagram will show the whole system and any external entities such as customers, suppliers, members, guests, etc.
- **To create a level 0 DFD, you should identify the external entities and the data flow between the external entities and the system**

- **This is the data flow represented and not physical objects, each data flow will only be in one direction**

- DFDs can also be one of the final stages of the analysis as they can be used to record the data flows that are currently taking place within an existing system
- At level 1, DFD shows the flow of data within part of the system or if the system is small then within the whole system
- If it's a part, then other parts of the systems are considered to be external entities
- Each flow of data must be linked to a process as something has to happen to the data before it can be stored or passed onto an external entity.
- **To create a level 1 DFD, first identify any external entities and any other parts of the system that will be classed as external entities. Identify the data flows to and fro those external entities**
- **Each data flow must have a process attached to it**
- **Data can't move directly from one external entity to another or from one data store to another or between an eternal entity and a data sore because a process is required to deal with the data**

Element	Purpose	Symbol
Data flow	This is the data that is flowing throughout the system.	
Process	This is an action that uses or manipulates data.	
Data store	This is a place where data is stored. This could be a hard disk, cloud storage or a paper file, for example.	
Data source or destination (inputs and outputs)	This is an external entity where the data originates or is destined.	
Duplication data source or destination.	This is where we have more than one source where data originates or is destined, to avoid crossing data flow.	

WORKED EXAMPLE 16.03

A hotel accepts online bookings for its hotel rooms. Guests make a booking online and the booking is received by the hotel. When the customer arrives, they are given an electronic key which includes encrypted data that will unlock the door and allow purchases at the bar. At the end of the stay, the guest is presented with a bill which must be paid before leaving.

CONTINUED

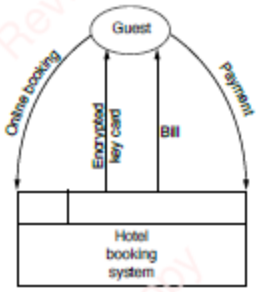


Figure 16.2: Level 0: DFD.

This DFD shows the system as the hotel booking system. It shows the guest as an external entity. It then shows the four items of data that flow between the guest and the booking system.

WORKED EXAMPLE 16.04

Figure 16.3 is a level 1 DFD for the hotel booking system. Any other aspects of the hotel system are considered to be external entities.

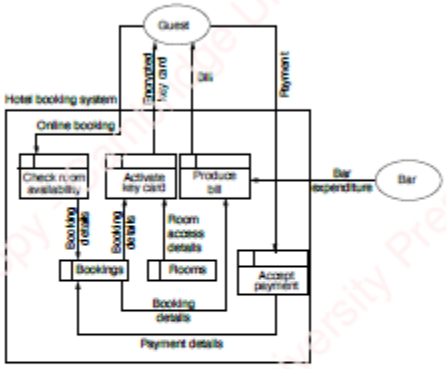


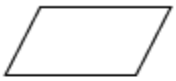
Figure 16.3: Level 1: DFD.

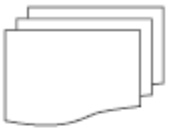
When the guest sends their online booking, the room availability is checked and the booking details are stored. When the guest arrives, their key card is activated by retrieving the room access details. When the guest is ready to leave, a bill is produced by retrieving the booking details and any bar expenditure from the bar system. When payment is made, the booking details are updated to confirm that payment has been made.

System Flowchart

- It shows the flow of data and processes within a complete system.
- System flowchart will show how various elements of an information system are related
- To create a system flowchart, identify the processes that takes place within the system then identify the different data files that will be used

- Use arrows to connect process to data files with the arrow pointing towards the ***data file*** if data is being stored or the arrow pointing towards the ***process*** if data is being retrieved
- If user input is required then add the ***manual input symbol*** at the appropriate place with the arrow pointing towards the ***process***.
- Identify any documents that are produced by the system and link each one with a process arrow pointing from the process to the document

Element	Purpose	Symbol
Manual input	Manual input into a computer system, for example using a keyboard.	
Arrow	Shows the direction of flow.	
Process	An activity within the system.	
Document	A single printed document.	

Element	Purpose	Symbol
Multi-document	Multiple printed documents.	
Magnetic disk	Storage on a magnetic disk, although in modern systems, other storage methods are likely.	
Magnetic tape	Data stored on a magnetic tape.	
Display	Output on a visual display.	

WORKED EXAMPLE 16.05

Figure 16.4 is a flowchart for taking a hotel booking online.

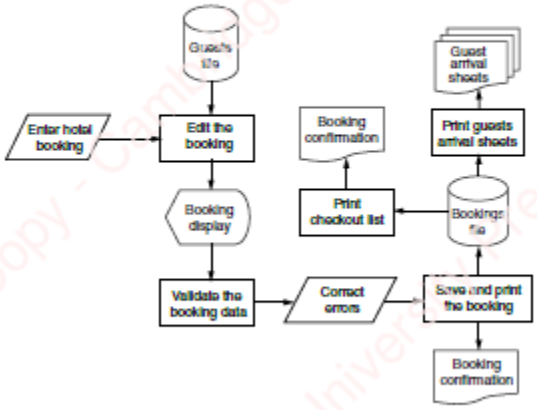


Figure 16.4: Online hotel booking flowchart.

CONTINUED

Hotel bookings are input manually using a keyboard. While editing the booking, the data that has been input is displayed on the screen. Data for the guest who will be staying at the hotel is retrieved from the guests' file. When the data has been input, it is validated and the user is required to correct any errors. A process then saves the booking and prints a booking confirmation document which can be sent to the guest. Each morning, guest arrival sheets, which include information about each guest and the room they will be staying in, are printed. Also in the morning, a checkout list is printed which shows which rooms are due to be vacated that morning.

Data Storage

Data dictionaries

- A data dictionary should be created to describe how data will be stored in tables within a database
- The fieldnames should be identified along with their data type, field size and format
- Primary keys and foreign keys should be identified including the names of tables to which the foreign keys link to
- Any input masks, validation rules or default values should be identified for each field along with an example of what typical data might look like
- An entity relationship diagram should be created to show the relationships between the tables that will be used.

WORKED EXAMPLE 16.06

This is an example of a data dictionary for a table about students.

Attribute	Data type	Length	Format	Validation Rule	Default Value	Sample Data	Primary Key	Foreign Key
Student ID	Integer	6	999999	Autonumber		459283	Yes	
Forename	Text	20	Xxxxx	Required		Mao		
Surname	Text	20	Xxxxx	Required		Zedong		
Other names	Text	50	Xxxxx	Required		Xi		
Tutor Group	Text	3	99X			12A		TUTOR GROUP – TgId
Date of Birth	Date	8	DD/MM/YYYY	<= 10 years before today		12/04/2007		
Date Joined	Date	8	DD/MM/YYYY	>= 10 years after date of birth		01/09/2018		
Date Left	Date	8	DD/MM/YYYY	> [Date Joined]				
Disability	Boolean	1		Required	False	True		

Files:

- Any files that will be used to import data should be designed, including the intended layout of data
- Any files generated by the system should be designed including the format in which the data will be exported

- The type of files should be specified, for example whether it is comma separated or tab delimited
- The contents of each column should be specified including the expected data type, length and format of each column

Input Forms

Data Collection Forms

- Data collection forms are documents that are used to collect data without the use of a computer such as questionnaires, application forms or reply slips
- It is important to design the form in such a way the required data can be collected

When designing a data collection form, it is good practice to follow these principles:

- avoid color as the document may not be printed in color
- include instructions about how to complete the form
- give clear instructions where the form should be returned
- identify which questions must be answered and which are optional
- provide enough space for each answer
- use tick boxes for multiple choice lists
- make it clear how many options are allowed to be chosen from a multiple choice list
- ensure all fonts are consistently used
- avoid cluttering the form with too much information or too many questions
- ensure the font style and size are legible
- if the respondent needs to complete a scale then explain what the scale represents (e.g. 1 = very dissatisfied, 10 = very satisfied)

Screen Layouts

- a screen can be used to ask the user for data to be input or to display information to the user or a mixture of bot

When designing a screen it is good practice to follow these principles:

- use color sparingly and appropriately, different colors could be used for questions and responses or for different types of data; colors that the users expects should be for example, green is seen as a positive color and red as negative color
- ensure all fonts are used with consistency
- avoid cluttering the screen with too much information but at the same time try to fit all information that needs to be viewed at the same time on a single screen
- ensure the font style and size are legible
- if the screen requires user input:
    - include instructions about how to complete the form
    - identify which questions must be answered and which are optional
    - provide enough space for each answer
    - use tick boxes for multiple choice lists that can have more than one answers
    - use drop down boxes or option buttons (radio buttons) for multiple choice lists that can have only one response
- when designing a screen or collection form, its only necessary to indicate where questions and responses will go, the types of response options and the styles to use, the layout of any information should also be indicated. The developer will then follow the design.

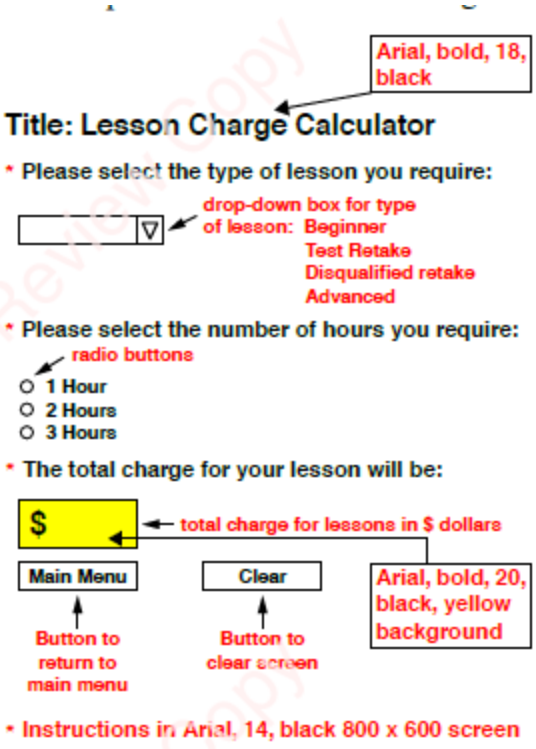

Arial, bold, 18, black
Title: Lesson Charge Calculator
* Please select the type of lesson you require:
drop-down box for type of lesson: Beginner
Test Retake
Disqualified retake
Advanced
* Please select the number of hours you require:
radio buttons
1 Hour
2 Hours
3 Hours
* The total charge for your lesson will be:
$
total charge for lessons in $ dollars
Main Menu
Clear
Arial, bold, 20, black, yellow background
Button to return to main menu
Button to clear screen
* Instructions in Arial, 14, black 800 x 600 screen

Figure 16.8: Example of a screen design.

Validation Routines

- They should be sued whenever possible and be appropriate in order to reduce the number of possible input errors
- They only need to be used for input data so any calculations or output data don't require validating
- Drop down boxes should always be used instead of lookup validation checks
- When designing a validation rule, identify the input data that is needed to be validated, the type of validation rule to be used, the rule that will be used and the error message that should appear if the data input is invalid
- Error messages should be positive and guide the user as to what to do to correct them

Input data	Validation type	Validation rule	Error message
Application number	Type	Must be a whole number	The application number must contain only numbers
Telephone	Length	Must be between 3 and 15 digits	Telephone number must be between 3 and 15 digits
Product code	Format	XX999XX9	The product code must be in the format XX999XX9 where X is a letter from A to Z and 9 is a number from 0 to 9

Table 16.4: Examples of a design for validation.

- Review Copy

Checking of data collected by forms

In addition to validation rules, any intended methods for verifying data input such as visual checking, double data entry or hash control should be specified (chapter 1 has more details)

Output Reports

Printed Copy Layouts

- When designing a printed copy layout, as well as following the same principles for output reports consideration needs to be made to the size of paper that will be used, the size of margins and intended audience
- Printed copies can often include tabular data

Output Screen Layouts

- The design of output screen layouts follows the same principal as input forms except that there is no need for instructions or input fields.

WORKED EXAMPLE 16.07

This example shows a report for listing all the products available.

Figure 16.9: Example of report listing for products.

1 The description, category and supplier data will be repeated for each record in the PRODUCTS table.

2 n = page number, nn = total number of pages

3 ## = total number of products in PRODUCTS table

In addition to these points, the size of paper, size of margins, font styles and sizes, colours and any gridlines should also be specified.

16.4 Development and testing

This is the stage where the design is implemented and tested

Test Plans

- Test plan is a detailed and structured plan of how testing should be carried out
- A test plan will identify what is being test, the type of test, the input data that should be used to test it, the expected result of the test and space to record the actual result.
- Each test is numbered

- All calculation works need to be identified and expected results determined and then tested. Any links or buttons used also need testing.
- A good test plan will provide a systematic outline of all features and functionality which will be continually updated to reflect any new risks as the software is developed

Number	Description	Type of test	Input data	Expected result	Pass/fail
4	Main menu button	Button	Click on main menu button on quote worksheet	The main menu worksheet opens	Pass
5	Clear button works	Button	Lesson type = advanced, number of hours = 2 Click on the clear button on quote worksheet	Lesson type = (blank) Number of hours = (blank)	Fail – number of hours remained as 2

Table 16.7: Example of a test plan for buttons.

The need for testing and purpose of test plan

- Testing is necessary because no programmer or developer is perfect and errors are expected which need to be found and rectified
- It is important to ensure the system is error free so the users can use the system knowing it's reliable and behaves as expected.
- Having a test plan ensures that all the pathways through a system and types of data have been tested to help minimize the number of errors
- A test plan will identify all the tests that are needed for every input, button, link, report, screen and all other elements of a system. It will include different type of test data including valid, invalid and extreme so that inputs are tested to their limits
- Without this planning, important parts of the testing would be missed out and errors could be left undiscovered
- The plan will cover all the users requirements and ensure that they are tested
- It will ensure all aspects of running a test plan are considered and prevent aspects being missed out
- It's also important so that tester know what the testing regime will involve and that there is a mechanism for ensuring each test is carried out as planned and signed off

Test Data

- it is the data that will be used for testing a system, it is a simulation of 'live data'.

- There should be enough test data generated to ensure that the system is able to cope under the pressure of large amounts of data in everyday use
- There will also need to be specific types of data to include to test different scenarios within the software including validation rules and queries

Type of test data	Description
Valid (also called normal data)	Data that will be accepted by the validation rule.
Invalid (also called abnormal or erroneous data)	Data that will not be accepted by the validation rule.
Extreme (also called extreme data)	Data that will only be accepted by the validation rule because it is at the limit of acceptability.

Table 16.5: Types of test data.

WORKED EXAMPLE 16.08

To test the validation rule that gender must be 'M' or 'F', the following test data could be used.

Type of test data	Data
Valid (normal)	M, F
Invalid (abnormal or erroneous)	B

To test the validation rule that a date must be between 1/1/2017 and 31/12/2017, the following test data could be used.

Type of test data	Test data
Valid (normal)	12/5/2017
Invalid (abnormal or erroneous)	15/7/2012, 4/6/2019
Extreme	1/1/2017, 31/12/2017

WORKED EXAMPLE 16.09

The following data for records could be used to test the query for males over the age of 50 (including 50).

Record number	Gender	Age	Reason
1	M	65	Both criteria met
2	F	25	Both criteria not met
3	M	25	Gender part met, age part not met
4	F	65	Gender part not met, age part met
5	M	50	Age part only just met
6	M	49	Age part only just not met

- Test data is also used to test queries
- Records will be created where there is data that meets the criteria of the query, doesn't meet the criteria and only just meets the criteria of the query and just fails to meet the criteria of the query
- Where there is more than a single criterion, data should be selected that only meets part of the criteria incase both pats have not been setup correctly

Alpha and Beta Testing

- Alpha testing is carried out by employees or the developer whereas beta testing is carried out by clients/end users/not employees
- Alpha testing uses a lab/test environment whereas beta testing does not require/use this/is in public
- Alpha testing is carried out at developers site whereas beta testing is carried out tester/end user location
- Aloha testing doesn't involve reliability testing whereas beta testing does involve
- Alpha testing involves both white and black box testing whereas beta testing usually involves only black box testing
- Critical issues can be fixed/addressed immediately during alpha testing whereas beta testing produces feedback on issues for use at later date/ in later versions
- Alpha testing ensures that the product is fit/ready for moving to be beta testing phase

White box and Black Box Testing

- White box testing is the testing the whole system in terms of structure and logic covering all paths through the system
- Black box testing is testing of inputs and outputs to a system or part of the system with no consideration for the working of the system
- White box testing is a testing method in which the internal structure/design/implementation is known to the tester
- Black box testing is a testing method in which the internal structure/design/implementation is NOT known to the tester
- In white box testing, programming knowledge is required, in black box testing it is not
- In white box testing, programming knowledge is in house, in black box testing it is not
- White box testing needs access to a detailed design specification, black box testing does not
- White box testing needs knowledge of the implementation, black box testing does not
- In black box testing only a limited number of test scenarios are actually performed due to not knowing how each module works but with white box testing each module can be tested in depth

- In black box test plans are difficult to design because so many different input data options have to be considered along with pathways through the system whereas with white box testing, test plans can be created for each calculation, navigation and input separately
- In white box testing the tester can identify the type of test data that will be required, particularly related to valid, invalid and extreme test data but with black box testing, the tester may not know what the boundary values are expected to be.

16.5 Implementation

Parallel Running

- Parallel running is when a new system ad an old system are run at the same time
- On an agreed date, the new system will become live but the old system will continue to run
- Data will need to be duplicated from the old system to the new system
- New data will need to be input into both systems and output will be produced from both systems
- This will continue until the organization is confident that the new system is running satisfactorily

Advantages:

- Less risky because if the new system fails the organization can continue to run using the old system
- The accuracy of the new system can be tested against the old system and any errors can be fixed

Disadvantages:

- Duplication of data input means additional staffing costs
- There may need to be additional hardware installed at the same time as the old hardware is still being used, which will require physical space
- Data may be input differently into the two systems, meaning that the data is not accurate in both

Direct Changeover

- Direct changeover is when a date is chosen for the old system to stop running and the new system to start running
- The systems do not run at the same time and there is a clear break from the old system to the new system
- Data will need to be transferred from the old system to the news system before the new system can be used

Advantages:

- This is cheap to implement because there is no duplication of work
- The data being used will be consistent because it is only being used in one system at a time
- There is no need for the new system to be compatible with the old system

Disadvantages:

- This is a risky method because any errors could lead to the system failing with no fallback
- All the training will need to be done in advance of changeover and so if there are a lot of users this could result in some forgetting what they have learned by the time they start to use the new system

Phased Implementation

- With phased implementation, parts of the new system will be introduced one at a time
- This often takes place when there is a large system with lots of functionality that can be easily separated into sections
- The olds system will run until a date has been agreed, at which point part of the olds system will be retired and part of the new system will start running
- After a while, another part of the old system will be retired and another part of the new system will start running. This will continue until the complete new system is fully running.

Advantages:

- If there are any errors they will only affect the part of the system that has changed over rather than the whole system
- End users can be trained how to use each phase of the new system and spend time using that phase before being trained in the next phase

Disadvantages:

- Delays can occur waiting for each phase to be running successfully before the net phase can start
- Users will be using two different systems and they may get confused as to which system they should be using of which part of their work, this could lead to data being updated in the wrong system.
- Both the old and new system need to be compatible with each other in order for data to be sued across both systems

Pilot Implementation

- Pilot organization takes place when art of an organization starts to use the new system while the rest of the organization continues to use the old system
- The new system is effectively being beta tested by the pilot group who may also be able to deliver training to the rest of the organization when the system goes fully live.

Advantages:

- If there are any errors, there will only affect the pilot group that are using the system
- Any errors found by the pilot group can be fixed before the system is installed for all users
- The pilot group can train other users on how to use the system because they will have experienced using it for real

Disadvantages:

- This is a slower method of changeover because the rest of the organization has to wait until the pilot has been completed satisfactorily
- Users in the pilot group might not be happy about using the new system while it may still have some errors in it and users not in the pilot group may be disgruntled that they have not been offered the opportunity to try the new software first

- Both the old and new system need to be compatible with each other in order for data to be shared between the pilot group and the users still using the old system

Choosing an implementation method

The most suitable changeover method will always be dependent upon the individual circumstances surrounding a new system. Factors that will need to be taken into account will include:

- How critical the system is to the running of the organization
- Cost
- Number of users in the organization
- The size of the new system

16.6 Documentation

User Documentation

- User documentation is a user guide giving instructions to the user on how to use the system.
- It can be printed or electronic form
- A printed user documentation should have a front cover that clearly identifies the name of the system and the whole guide should have a header and a footer with page numbers
- An electronic version would include hyperlinks to those pages
- An introduction to the purpose of the user guide should be included but it only needs to be a few sentences
- All the software and hardware requirements will be listed within the guide
- The main part of the user guide will be the instruction on how to use the system
- This should include written instructions together with screenshots of the system or photographs of the hardware
- Arrows can be used to point to parts of the screenshots or photographs

- Bullets or numbering should be used to break instructions down into manageable tasks
- A glossary to show an alphabetical list of any technical terms that have been used within the user guide and a definition of each of these terms
- There should be a troubleshooting sections that includes a table of common problems(for example, (error messages)together with a description of possible solutions for overcoming the problem
- An index will be included at the end of the user guide with page numbers for each popular term

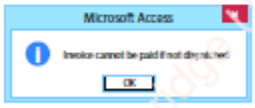
WORKED EXAMPLE 16.11

This is an example of a troubleshooting guide for a printer.

Problem	Cause	Solution
Orange light displayed on printer.	No paper in feeder tray.	Add paper to the feeder tray.
Red light displayed on printer.	Paper is jammed in the printer.	Open the paper feeder tray and check there is no paper stuck there. Open the back door and check there is no paper stuck there. Open the toner door, remove the toner and check there is no paper stuck there. If any paper is found, gently pull any paper that is stuck.
Error message on computer says 'Printer Offline'.	Printer is turned off. Printer is not connected to the computer.	Turn on printer. Ensure the USB cable is connected between the computer and the printer.

CONTINUED

Below is an example of a troubleshooting guide for an order processing system.

Problem	Cause	Solution
Figure 16.10: Error message 1.	On the New order form you have specified a quantity larger than the amount currently in stock.	Enter a smaller quantity.
 **Figure 16.11:** Error message 2.	On the Order details form you have ticked the Paid box before the Invoice has been dispatched.	Check the invoice has been dispatched. If it has, then tick the Invoice dispatched box.

Why User Documentation Is Needed:

- It is needed so that the user can learn how to use the new system or lookup how certain features are supposed to work

- The troubleshooting section will be importatn for the user to understand wha has casued an eror to occur and what they need to do in ordeer to stop the error from occuring.

Technical Documentation

- Technical documentation is an overview of the structure of the system, how it was put together and how it works
- It will include a data dictionary to show how data has been structured within the system
- Any programming code or macros will be annotated to explain their purpose and anything unusual along with a list of variable including their data types and purposes
- All validation rules will be listed with the criteria used for successful input and the error messages that they generate
- The purpose of calculations within the system will be identified and an explanation given on how each calculation works
- All buttons and links will be listed and their purpose identified
- The technical documentation will also include flowcharts and to show different parts of the system work and other diagrams that may have been used during design and development such as entity relationship diagrams and screen connectivity diagrams

There will be an installation guide for the installation guide for the installation team and if the system has to be installed again in future. All the results of the testing will be recorded, including any known errors and bugs. Backup routines were configured and how to restore from a backup. All the security settings will be documented to show which groups have access to each pat of the system and the permissions they have been granted. The software and hardware requirements will also be listed.

Why Technical Documentation Is Needed:

- It is required so that anybody carrying out future maintenance on the system will be able to understand how the system was developed and how it is configured
- It is unlikely that the person carrying out future maintenance is part of the original development team and so will not be familiar with the system without the technical documentation
- Even if it's the same person or team carrying out the maintenance, they will need the documentation to remember the structure of the system.

16.7 Evaluation

It is a formal review of the project

Methods of Evaluating a New System:

In terms of efficiency, ease of use and meeting user requirements

- When a system has been developed and installed, the whole process will be evaluated, this is sometimes referred to as a review
- The evaluation considers how the project team and end users worked together so that lessons can be learned for future projects
- Users will be given questionnaires to find out what they think of the new system and how they feel its improving their workflow or not
- Selected users will be given specific tasks to complete that will be observed to see whether the new system is up to their expectations
- Some users will be interviewed about their interaction with the new system
- Users will also be asked if the system meets the user requirements, each requirement will be considered in turn to determine if it has been fulfilled
- If it hasn't been fulfilled then actions will be set to rectify the situation in the long run.
- The efficiency of the new system will also be discussed
- Users will be given an opportunity to feedback on how well the new system is working for them and if there are any problems
- It is expected that the new system will work more efficiently than the old system. However if there are problem that need addressing, then actions will be taken
- The requirements specification should have specified how easy the new system should be to use. This can be rather subjective so it is difficult to measure
- Again feedback will have been gained from users as to how well they have adapted to the new system and how easy or not they find it to use now they are using it regularly
- If there are issues regarding ease of use then plans can be made to simplify any processes by additional features to the software if necessary
- There will also be an opportunity for users to make suggestions or future improvements or additions to the system.

A new computer application has been developed to replace the current application that is in use by a company. Describe how the new application will be evaluated [6]

- Determine if the new application runs/operates efficiently by saving time/resources compared to the old application
- Determine if the new application is easy to use/can be understood by users with little extra training
- Compare with requirement specification to check that all requirements have been addressed/go through requirements one by one
- Determine if the new application actually meets the needs of the company
- Gather feedback from users of the application using questionnaires/interviews/observation techniques
- Check that users like the application/find the application easier to use/check whether users can suggest improvements
- Limitations/problem/issues of the new application are identified
- System/application development cycle is started again to correct/redesign/test and implement improvements

16.8 Maintenance

Maintenance takes place after the system has been delivered to a customer and it is being used. It is the changes made to a system after its implementation.

Perfective Maintenance

- It enhances/improves/increases the performance (of software) after delivery/during the lifetime of the software
- It enhances/improves/increases the usability/user experience of the software
- It enhances/improves/increases the reliability/security of the software (to increase its life span)
- It enhances/improves/increases the (ease of) maintenance of the software

Why is it needed:

- To be always looking for ways to improve a system
- To make the system more updated and relevant that could be possible due to new technology
- To prevent it from being inefficient and outdated
- To implement new users ideas that are likely to improve ease of use and enhance efficiency
- Eg: an online accounts application sends out automatic reminders to customers when payments haven't been made. These reminders are sent to a single customer contact. The system only allows for the contact details of one person to be stored. Many users of the accounts application have requested that the system be adapted to store details of multiple contacts for each customer and that contacts who should receive invoice payment reminders are identified within the software so that they go to the right person.

Adaptive Maintenance

- It (modifies and) updates software after delivery to the users
- In response to new/up to date technology/hardware/operating environment
- In response to changes in industry/business requirements
- In response to changes/amendments to regulations/legislation

Why is it needed:

- Systems need to adapt to changes due to internal procedures, new laws, new technology to ensure it works effectively and doesn't produce incorrect errors
- Eg: For example, the government introduced new requirements for organizations to provide pensions to all employees. The online accounts application needed to be updated to include a facility for checking that all employee pay slips include pension payments unless they have opted out of the scheme.
- Eg: The accounting software was supposed to show a paper clip symbol to indicate that a receipt had been uploaded against a recorded expenditure. When a web browser was upgraded, this paper clip stopped being displayed. The software had to be adapted to work with the new web browser.

Preventive Maintenance

- (Modifies and) updates software after delivery to the users
- To avoid/prevent/protect against possible problems/faults/errors that might occur in the future
- To fix minor/insignificant/errors that do not affect function/cosmetic errors (but may become significant in the future)

Why is it needed:

- to prevent problems arising within a system (both hardware and software)
- Eg: Hardware should be regularly cleaned to stop dust from blocking any fans and regular scans of storage media should be carried out to ensure they are working properly.
- Eg: Heat should be monitored within systems for any abnormalities to prevent hardware failures.
- Eg: Data should be regularly checked for consistency and integrity. Performance of the system should be monitored to ensure that the processor, memory and storage are all working efficiently.
- By carrying out regular preventative maintenance, system downtime can be avoided.

Corrective Maintenance

- It corrects a problem/issue in the system that have been identified in error reports from users
- After the system has already broken down/failed/not working properly
- It restores the functioning/operability of the system by replacing components/adjusting code

Why is it needed:

- To ensure system runs efficiently and accurately and produce results required by an organization.

- To remove bugs that cause a system to slow down or crash it, reducing overall productivity and causing frustration to user
- Eg: a graphics application would intermittently stop responding for several seconds, this was not supposed to happen and so corrective maintenance was required

Describe the steps that the technician should take when carrying out corrective maintenance [3 marks]

- Diagnosing the problem (by testing system modules/components)
- Ask users for/gather information about the error/problem
- Identifying the problem/error/fault in the system
- Removing the faulty component/isolating the faulty code/module
- Replacing the faulty component with a new one (and testing it)
- Updating/amending the faulty/problematic software code/module
- Check for and removing viruses/malware/uninstalling harmful programs
- Reformatting storage devices and perfume a system restore
- Refer to technical documentation
- Take notes/make a report for reference
- Retest the system at the end of process

How maintenance is carried out:

- A modification request (or problem report) will be generated which could be a request by a user or it could be as a result of collating error logs from the system
- The modification request must clearly describe the existing functionality of the system and the desired functionality of the system
- The modification request will be classified as either perfective, adaptive, preventive or corrective maintenance
- Diagnosing the problem (by testing system modules/components)
- Ask users for/gather information about the error/problem
- Identifying the problem/error/fault in the system
- Removing the faulty component/isolating the faulty code/module
- Replacing the faulty component with a new one (and testing it)
- Updating/amending the faulty/problematic software code/module
- Check for and removing viruses/malware/uninstalling harmful programs

- Reformatting storage devices and perfume a system restore
- Refer to technical documentation
- Take notes/make a report for reference
- Retest the system at the end of process

^Adjust points according to type of maintenance asked

16.9 Prototyping

- A prototype is a mockup of a software solution in a primitive form which is used during the design stage to demonstrate how a system will look and work.
- It is usually focused on the user interface rather than any data structure
- It is used to let a client get a feel for the new system before it is developed and can provide feedback that can be acted upon
- The client is also able to compare the prototype against the requirements specification and it also allows the client the opportunity to explain their requirement more clearly having seen the designers interpretation

Evolutionary q11

- It follows a similar pattern to incremental prototyping in that it is iterative in nature however there are nor requirements specification at the start of the project but instead there might be a goal or an aim
- It involves creation of a basic system prototype, followed by successive refinement based on user feedback and changing requirements
- The process allows stakeholders/client to interact with an evolving prototype early in development, facilitating better communication and understanding.
- Through multiple iterations, the prototype gradually converges towards the final product, enabling flexibility, adaptability to evolving needs and early detection and resolution of issues
- This approach is particularly beneficial for projects where user involvement, adaptability and risk reduction are critical factors

Advantages:

- The users can get used to using parts of the system before having to use the whole system, which will reduce the need of bulk training
- There's early user involvement, providing a tangible prototype early in development process promotes better communication and understanding of the requirements
- The iterative nature allows adaptability based on continuous changing requirements and incorporation of new features based on ongoing feedbacks
- Issues can be identified and addressed early which reduces the likelihood of defect in the in the final product
- Reduced risk of misunderstanding
- Improved stakeholder satisfaction

Disadvantages:

- The iterative nature makes it more time and resource intensive compared to other development prototypes
- Continuous changes and additions may lead to scope creep where the project expands beyond its initially defined boundaries impacting budget and timeline
- Frequent changes can make it more challenging to maintain comprehensive documentation which may be essential for long term maintenance and understanding the system
- Not suitable for large, complex projects with well-defined requirements where a more predictive and structured approach is preferred
- Success depends on the availability and active participation of users for providing timely feedback which may not always be feasible in some projects

Incremental q41

- Incremental takes an iterative approach, in that requirements are specified, an initial prototype is developed, the prototype is reviewed and then requirements are clarified and the prototype is improved on feedback
- Project/model/product is broken into several/many sub projects/models/products
- Model/product is partly built/built on previous iterations
- Top priority/essential requirements are produced first and completed
- Testing occurs at each iteration/incremental stage
- At each stage, only clear understood requirements are developed

- As soon as a requirement is produced/testing is complete/its development is stopped/goes no further
- Each prototype can be functional and if required can be used the client until the next iteration of the prototype is ready.
- This allows end users to get new or enhanced features at the end which they wouldn't have envisaged at the initial requirements stage

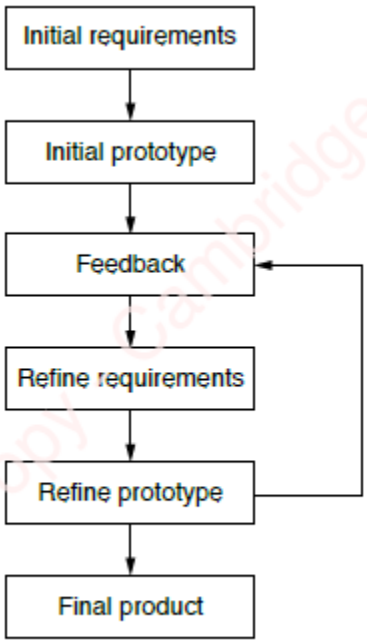


Figure 16.13: Iterative prototyping.

Advantages:

- Each stage is an identifiable milestone to track progress
- Testing is carried out at each incremental stage, allowing faults/problems to be identified/rectified quickly/before final completion
- Testing is easier/quicker because changes are usually small at each iteration of product
- Client/user can see/use new features and review these for any further changes
- Product delivery /completion of early/working versions is faster than other methods of development
- Risks can be managed better since risks are identified and handled during each iteration/incremental stage
- Product delivery/completion of early/working versions can cost less than other methods of development

Disadvantages:

- Incremental/additional changes may affect functionality of earlier versions in an unexpected manner
- May raise problems at the end because all requirements are not produced/created/identified in product at start
- Overall cost of many versions/increments can become exorbitant/excessive

Throw-Away/Rapid

- It is also called rapid prototyping, in it the prototype never becomes the part of the final delivered software and is discarded
- A loosely working model is created following a short investigation with aim being to get something tangible to the client as soon as possible for feedback as to check how well the requirement s are being met
- It allows the requirements to be fine-tuned early in the process which is more cost effective than trying to make changes later when considerable work has been done
- The main aspect to the prototype will be user interface which the client will be able to test and experience, it (user interface) will appear to work by being simulated

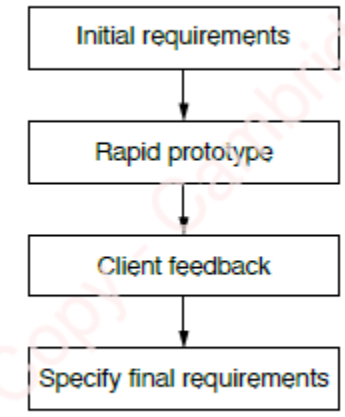


Figure 16.14: Throwaway prototyping.

Advantages:

- Allows quick development and rapid progression of the project

- Useful for validating design concepts and ideas early in the development process before committing to full scale misunderstandings
- It is cost effective as compared to others
- Identifying issues and refining requirements in the early stages helps mitigate the risk of developing a system that doesn't meet user needs or expectations.

Disadvantages:

- When users see what looks like a working interface with a throwaway prototype, they don’t realize how much more effort is required to make it into a working solution and may have false expectations as to the timescale
- The iterative process of feedback can sometimes last too long if the user is regularly wanting changes to be made to the latest prototype
- There’s limited reusability since throwaways are not intended for use in final product
- There are documentation challenges as quick prototyping may make it challenging to transfer knowledge to subsequent development phases
- Rapid prototyping may lead to the inclusion of unnecessary features or scope creep if not carefully managed, impacting project timelines and resources.
- Throwaway prototyping may be less suitable for large, complex projects with well-defined requirements, where a more structured and incremental approach might be preferred.

Advantages	Disadvantages
Problems can be identified early during the process and modifications made before it becomes very costly to make changes.	Requirements analysis can be rushed, meaning that prototypes don't reflect much of what the end users were expecting.
Requirements can be clarified and refined following feedback on the prototypes.	With rapid prototyping, the prototype can become rushed and, when trying to develop it into a working system, it may have significant design flaws or structural errors that carry through to the end solution.
The end users will be involved more in the process, giving them more ownership of the solution and providing valuable feedback.	When users see the prototype, they can often get lots of new ideas about features they would like to be included, which can lead to disappointment if these features can't be funded. This is known as 'feature creep'.
If the prototype is evolutionary, then users can get used to using parts of the system before having to use the whole system, which will reduce the need for bulk training. It's much cheaper to make changes earlier in the process than after real development has taken place.	When users see what looks like a working interface with a throwaway prototype, they don't realise how much more effort is required to make it into a working solution and may have false expectations as to the timescale. The iterative process of feedback can sometimes last too long if the user is regularly wanting changes to be made to the latest prototype.
By listening to feedback from end users, the developers will have a much better understanding of what the users are expecting and so a better quality solution will be provided.	The initial costs of developing a prototype are high compared with traditional designs.

Table 16.9: Advantages and disadvantages of prototyping.

16.10 Methods of software development

Agile

- The agile approach to software development is able to respond to a customer's changing requirements even late in the development life cycle
- It is expected that the customers requirements will evolve as the project develops

- The process can cope with change and harness that change for the benefit of the customer
- Individuals and the interaction between people is more valued than design process and tools. It is also focuses more a working software to be developed than a comprehensive supporting documentation
- It uses iterations with each iteration including the planning, design, development and testing part of the software
- At the end of an iteration, a working solution for that part of the software is ready of demonstration
- Several iterations are usually required before a product is ready to be released or updated
- As an iteration include designers, developers and tester they are expected to be in the same place
- This ensures a collaborative approach to each iteration over an contract negotiation
- Agile is both iterative and incremental, its incremental because each iteration can be planned to be improved upon in future iteration

Advantages:

- The customer gets to see completed parts of the software quickly which enables them to adapt their requirements for other parts of the software
- The customer, designers, developers and testers are constantly interacting with each other which reduces delay
- If the business needs aren't fully known at the stat of the project then they can evolve as the project progresses
- There is unlimited access to the customer which means their needs are more likely to be met
- There is no complicated bureaucracy that delays decision making
- The customer doesn't have fixed budget or timescale so customer can be met above budget or timescale
- Parts of software can be deployed when they are ready so the customer sees the value of that software sooner
- Bugs and problems can be fixed quickly because the designers, developers and testers for each iteration are working closely together

Disadvantages:

- As requirements change, it's difficult to know how much a project is going to cost and how long it will take
- The project can easily go off track if requirements change too much
- Experienced senior programmers are needed to be able to make quick decisions during each iteration
- There will be minimal documentation produced which means new team members can take longer to get up to speed with the project and there can be a larger learning curve when maintaining the software
- Projects may seem to never have an end in sight
- Not having a fixed budget or timescale creates a lot of uncertainty for the client and is not a popular move with senior members of an organization
- The software, especially the user interface, can become disjointed because each iteration is worked on separately

Iterative

- Iterative development is creating a system or adding new functionality in a repetitive cycle
- It is about planned rework throughout a project
- It allows for feedback to be given by the customer and improvements to be made based on the feedback
- It usually involves building a bit of every part of a solution before seeking feedback and then revisiting the whole solution to make improvement

Advantages:

- Flexibility and adaptability
- Early delivery of core functionality
- Continuous feedback
- Risk mitigation
- Improved quality

Disadvantages:

- Potentially extended timelines
- Resource intensive
- Documentation challenges
- Dependency on user availability
- Management complexity

Incremental

- Incremental development is creating a system or adding new functionality in small parts
- It is about development in phases
- Part of the software is developed fully, including improvements from feedback before moving on to the next part of the software

Advantages:

- Early delivery of partial functionality
- Increased flexibility
- Easier to manage and test
- Reduced risk of project failure
- User involvement and feedback

Disadvantages:

- Complexity in integration
- Increased management overhead
- Incomplete features
- Dependency on initial architecture
- Potential for scope creep

Rapid Application Development (RAD)

- Determine user/system requirements/aim of project
- Create early prototype(s) that are (partly) functionally very quickly
- Gather user feedback on early prototypes
- Use feedback to inform prototyping to create high quality prototype
- Repeat steps/iterate prototyping and user feedback stages until product software is finished
- Test prototypes/software throughout development
- Create user/technical documentation as required
- Present/produce final product for rollout to users

Advantages:

- The high level of user involvement means that the end solution is more likely to be suitable for the end users who will also have ownership of the solution
- Users are often not sure of what the requirements of a system should be from the outset so the evolutionary approach of RAD enables the requirements to evolve
- As users are involved throughout the whole project, it is quickly recognized when a requirement is over ambitious and therefore the requirement can be simplified or removed at an early stage
- The strict deadlines ensure that the project will be completed on time and prevent feature creep
- Prototyping of the interface with the user involvement means less time is spent on design and more on development leading to a shorter overall project
- Software application frameworks mean that a user interface can be developed quickly and users can even be involved in configuring the layout of screens and reports
- More likely to complete within budget

Disadvantages:

- Requirements are not clearly specified from the outset and so the final solution may be different
- Users are required throughout the whole process. This can lead to work overload or the need for temporary staff
- The strict deadlines mean that some parts of the project could be rushed and not competed to a high level quality

- Existing software modules will not have been designed for the exact requirements of the system
- Software application framework don't produce particular efficient code and so the end solution will not run as quickly as if it had been developed from scratch
- Users who aren't involved in the rad approach may be disappointed that they didn't have a say in the process and the system may not meet their specific needs

Waterfall

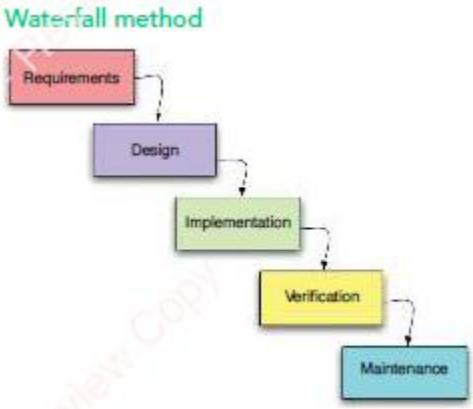


Figure 16.15: Waterfall method.

- It is an iterative development model
- It involves the creation of system and software requirements for the application at the beginning of the project.
- There will be considerable communication with user to elicit the required details and expectation and analysis will take place to create models, schemes and business rules
- During design stage, the technical designs/design specifications are created including e.g language
- During implementations stage, the code is implemented by writing the code/units/integration of units of software
- Creation of technical/user documentation
- Testing using a test plan to discover and correct errors
- Verification phase takes place to ensure the project meets the customers' requirements
- Deploying the software by the installation, mitigation, support and maintenance of the finished product

Advantages:

- Problems can be found and fixed early in the processes
- Each stage is completed before moving onto the next stage making management o the project more structured and controllable
- It's a simple model to understand and use because each stage is distinctly separate from the other stages
- It works well for projects where the requirements are clearly understood and are unlikely to change
- The process followed and the resulting software are well documented, meaning new team members can get up to speed very quickly
- The client and project team work to a set timescale and a set budget which removes a lot of uncertainty from projects
- The requirements are clearly set from the beginning of the project and are used to measure the project's success when it's completed

Disadvantages:

- The customer only sees the product at the end of the project
- Its hard to measure progress within each stage because the milestones are just the beginning and end of each stage
- It doesn't work well from comes projects where the requirement cover a large variety of aspects and aren't clearly understood by everyone involved
- Projects can take longer to delivery than other methods because of the emphasis on planning and documentation
- Client involvement is limited to set times within the project which can lead to requirements not being met to the clients expectation
- The model doesn't accommodate changes to requirements